DGrok Delphi Grammar

DGrok is my project to write a parser for the Delphi language, and then to build interesting tools on top of that parser. For more information on DGrok, see the DGrok posts on my blog.

This page shows the Delphi grammar as I've puzzled it out so far, and indicates how much of it I've written a working parser for. Solid underline means that rule is completely implemented in my parser; broken underline means partly implemented; no underline means something I haven't started yet.

Current status: Completed 98 rules of 98 = 100% complete.

Last updated Wed Oct 03 22:48:00 Central Daylight Time 2007.

— Joe White

Table of Contents

AddOp	ExceptionItem	GotoStatement	OpenArray	RecordType	UnaryO
ArrayType	ExportsItem	Ident	Package	RelOp	Unit
AssemblerStatement	ExportsSpecifier	IdentList	PackedType	RepeatStatement	UsedU
AssemblyAttribute	ExportsStatement	IfStatement	Parameter	RequiresClause	UsesCl
Atom	Expression	ImplementationDecl	ParameterExpression	SetLiteral	VarDec
BareInherited	ExpressionList	ImplementationSection	ParameterType	SetType	Varian
Block	ExpressionOrAssignment	InitSection	ParenthesizedExpression	SimpleExpression	Varian
CaseSelector	ExpressionOrRange	InterfaceDecl	Particle	SimpleStatement	VarSec
CaseStatement	ExpressionOrRangeList	InterfaceSection	PointerType	Statement	Visibili
ClassHelperType	ExtendedIdent	InterfaceType	PortabilityDirective	StatementList	Visibili
ClassOfType	Factor	LabelDeclSection	ProcedureType	StringType	Visibili
ClassType	FancyBlock	LabelId	Program	Term	WhileS
ConstantDecl	FieldDecl	MethodHeading	Property	TryStatement	WithSt
ConstSection	FieldSection	MethodImplementation	PropertyDirective	Type	
Directive	FileType	MethodOrProperty	QualifiedIdent	TypedConstant	
EnumeratedType	ForStatement	MethodReturnType	RaiseStatement	TypeDecl	
EnumeratedTypeElement	Goal	MulOp	RecordHelperType	TypeSection	

Likely Targets (0)

These rules look like their dependencies are met, and are likely targets to be implemented next. Numbers in parentheses show how many places the rule is used.

AddOp [^]

Backlinks: SimpleExpression

(Completed) -> '**+**'
(Completed) -> '**-**'
(Completed) -> OR
(Completed) -> XOR

ArrayType [^]

Backlinks: Type

(Completed) -> ARRAY ['**[**' (Type ['**,**'])+ '**]**'] OF Type

AssemblerStatement [^]

Backlinks: Block

(Completed) -> ASM
(Completed) <assemblylanguage>
(Completed) END

AssemblyAttribute [^]

Backlinks: ImplementationDecl, Package

(Completed) -> '**[**' ASSEMBLY '**:**' Expression '**]**'

Atom [^]

Backlinks: Factor

(Completed) -> Particle
(Completed) ('.' ExtendedIdent
(Completed) | '[' ExpressionList ']'
(Completed) | '^'
(Completed) | '(' (ParameterExpression [','])* ')'
(Completed))*

BareInherited [^]

Backlinks: SimpleStatement

(Completed) -> INHERITED

Block [^]

Backlinks: FancyBlock, InitSection, SimpleStatement

(Completed) -> BEGIN [StatementList] END
(Completed) -> AssemblerStatement

CaseSelector [^]

Backlinks: CaseStatement

(Completed) -> (ExpressionOrRange [','])+
(Completed) ':' [Statement] [';']

CaseStatement [^]

Backlinks: SimpleStatement

(Completed) -> CASE Expression OF
(Completed) (CaseSelector)+
(Completed) [ELSE [StatementList]]
(Completed) END

ClassHelperType [^]

Backlinks: Type

(Completed) -> CLASS HELPER
(Completed) ['(' QualifiedIdent ')']
(Completed) FOR QualifiedIdent
(Completed) (VisibilitySection)*
(Completed) END

ClassOfType [^]

Backlinks: Type

(Completed) -> CLASS OF QualifiedIdent

ClassType [^]

Backlinks: Type

(Completed) -> CLASS
(Completed) [ABSTRACT | SEALED]
(Completed) ['(' (QualifiedIdent [','])+ ')']

The remainder is optional, but only if the base class is specified and lookahead shows that the next token is a semicolon

(Completed) (VisibilitySection)*
(Completed) END

ConstantDecl [^]

Backlinks: ConstSection

(Completed) -> Ident
(Completed) [':' Type]
(Completed) '=' TypedConstant
(Completed) (PortabilityDirective)*
(Completed) ';'

ConstSection [^]

Backlinks: ImplementationDecl, InterfaceDecl, VisibilitySectionContent

(Completed) -> (CONST|RESOURCESTRING)
(Completed) (ConstantDecl)+

Directive [^]

Backlinks: MethodHeading, ProcedureType

(Completed) -> [';'] ABSTRACT
(Completed) -> [';'] ASSEMBLER
(Completed) -> [';'] CDECL
(Completed) -> [';'] DISPID Expression
(Completed) -> [';'] DYNAMIC
(Completed) -> [';'] EXPORT
(Completed) -> [';'] EXTERNAL [Expression (ExportsSpecifier)*]
(Completed) -> [';'] FAR
(Completed) -> [';'] FINAL
(Completed) -> [';'] FORWARD
(Completed) -> [';'] INLINE
(Completed) -> [';'] LOCAL
(Completed) -> [';'] MESSAGE Expression
(Completed) -> [';'] NEAR
(Completed) -> [';'] OVERLOAD
(Completed) -> [';'] OVERRIDE
(Completed) -> [';'] PASCAL
(Completed) -> [';'] REGISTER
(Completed) -> [';'] REINTRODUCE
(Completed) -> [';'] SAFECALL
(Completed) -> [';'] STATIC
(Completed) -> [';'] STDCALL
(Completed) -> [';'] VARARGS
(Completed) -> [';'] VIRTUAL
(Completed) -> [';'] PortabilityDirective

EnumeratedType [^]

Backlinks: Type

(Completed) -> '(' (EnumeratedTypeElement [','])+ ')'

EnumeratedTypeElement [^]

Backlinks: EnumeratedType

(Completed) -> Ident ['=' Expression]

ExceptionItem [^]

Backlinks: TryStatement

(Completed) -> ON
(Completed) [Ident ':']
(Completed) QualifiedIdent DO
(Completed) [Statement]
(Completed) [';']

ExportsItem [^]

Backlinks: ExportsStatement

(Completed) -> Ident (ExportsSpecifier)*

ExportsSpecifier [^]

Backlinks: Directive, ExportsItem

(Completed) -> (INDEX | NAME) Expression

ExportsStatement [^]

Backlinks: ImplementationDecl

(Completed) -> EXPORTS (ExportsItem [',']) + ';'

Expression [^]

Backlinks: AssemblyAttribute, CaseStatement, Directive, EnumeratedTypeElement, ExportsSpecifier, ExpressionList, ExpressionOrAssignment, ForStatement, IfStatement, InterfaceType, Parameter, ParameterExpression, ParenthesizedExpression, PropertyDirective, RaiseStatement, RepeatStatement, StringType, TypedConstant, VarDecl, WhileStatement

(Completed) -> SimpleExpression (RelOp SimpleExpression)*

ExpressionList [^]

Backlinks: Atom, VariantGroup, WithStatement

(Completed) -> (Expression [','])+

ExpressionOrAssignment [^]

Backlinks: SimpleStatement

(Completed) -> Expression
(Completed) -> Expression ':=' Expression

ExpressionOrRange [^]

Backlinks: CaseSelector, ExpressionOrRangeList, Type

(Completed) -> SimpleExpression ['..' SimpleExpression]

ExpressionOrRangeList [^]

Backlinks: SetLiteral

(Completed) -> (ExpressionOrRange [','])+

ExtendedIdent [^]

Backlinks: Atom, QualifiedIdent

(Completed) -> Ident
(Completed) -> <keyword>

Factor [^]

Backlinks: Factor, Term

(Completed) -> Atom
(Completed) -> UnaryOperator Factor

FancyBlock [^]

Backlinks: MethodImplementation

(Completed) -> (ImplementationDecl)*
(Completed) Block

FieldDecl [^]

Backlinks: FieldSection, VariantGroup

(Completed) -> IdentList ':' Type (PortabilityDirective)* [';']

FieldSection [^]

Backlinks: VisibilitySectionContent

(Completed) -> [[CLASS] VAR]
(Completed) (FieldDecl)*

FileType [^]

Backlinks: Type

(Completed) -> FILE
(Completed) -> FILE OF QualifiedIdent

ForStatement [^]

Backlinks: SimpleStatement

(Completed) -> FOR Ident ':=' Expression (TO | DOWNTO) Expression DO [Statement]
(Completed) -> FOR Ident IN Expression DO [Statement]

Goal [^]

(Completed) -> Program
(Completed) -> Package
(Completed) -> Unit

GotoStatement [^]

Backlinks: SimpleStatement

(Completed) -> GOTO LabelId

Ident [^]

Backlinks: ConstantDecl, EnumeratedTypeElement, ExceptionItem, ExportsItem, ExtendedIdent, ForStatement, IdentList, LabelId, MethodHeading, Particle, Program, Property, QualifiedIdent, TypeDecl, Unit, UsedUnit, VariantSection

(Completed) -> <identifier>
(Completed) -> <semikeyword>
(Completed) -> '&' <identifier>
(Completed) -> '&' <semikeyword>
(Completed) -> '&' <keyword>

IdentList [^]

Backlinks: FieldDecl, Parameter, Program, VarDecl

(Completed) -> (Ident [',])+

IfStatement [^]

Backlinks: SimpleStatement

(Completed) -> IF Expression THEN [Statement]
(Completed) [ELSE [Statement]]

ImplementationDecl [^]

Backlinks: FancyBlock, ImplementationSection, Program

(Completed) -> LabelDeclSection
(Completed) -> ConstSection
(Completed) -> TypeSection
(Completed) -> VarSection
(Completed) -> MethodImplementation
(Completed) -> ExportsStatement
(Completed) -> AssemblyAttribute

ImplementationSection [^]

Backlinks: Unit

(Completed) -> IMPLEMENTATION
(Completed) [UsesClause]
(Completed) (ImplementationDecl)*

InitSection [^]

Backlinks: Program, Unit

(Completed) -> END
(Completed) -> Block
(Completed) -> INITIALIZATION
(Completed) [StatementList]
(Completed) [FINALIZATION
(Completed) [StatementList]]
(Completed) END

InterfaceDecl [^]

Backlinks: InterfaceSection

(Completed) -> ConstSection

(Completed) -> TypeSection
(Completed) -> VarSection
(Completed) -> MethodHeading

InterfaceSection [^]

Backlinks: Unit

(Completed) -> INTERFACE
(Completed) [UsesClause]
(Completed) (InterfaceDecl)*

InterfaceType [^]

Backlinks: Type

(Completed) -> (INTERFACE | DISPINTERFACE)
(Completed) ['(' QualifiedIdent ')']
(Completed) ['[' Expression ']']
(Completed) (MethodOrProperty)*
(Completed) END

LabelDeclSection [^]

Backlinks: ImplementationDecl

(Completed) -> LABEL (LabelId [','])+ ';'

LabelId [^]

Backlinks: GotoStatement, LabelDeclSection, Statement

(Completed) -> <number>
(Completed) -> Ident

MethodHeading [^]

Backlinks: InterfaceDecl, MethodImplementation, MethodOrProperty

(Completed) -> [CLASS]
(Completed) (PROCEDURE | FUNCTION | CONSTRUCTOR | DESTRUCTOR | OPERATOR)
(Completed) QualifiedIdent
(Completed) (
(Completed) ['(' (Parameter [';'])* ')']
(Completed) [':' MethodReturnType]
(Completed) (Directive)*
(Completed) | '=' Ident
(Completed))
(Completed) [';']

MethodImplementation [^]

Backlinks: ImplementationDecl

If the MethodHeading does not include 'external' or 'forward':

(Completed) -> MethodHeading
(Completed) FancyBlock ';'

If the MethodHeading does include 'external' or 'forward':

(Completed) -> MethodHeading

MethodOrProperty [^]

Backlinks: InterfaceType, VisibilitySectionContent

(Completed) -> MethodHeading
(Completed) -> Property

MethodReturnType [^]

Backlinks: MethodHeading, ProcedureType, Property

(Completed) -> QualifiedIdent
(Completed) -> STRING

MulOp [^]
Backlinks: Term

(Completed) -> '*'
(Completed) -> '/'
(Completed) -> DIV
(Completed) -> MOD
(Completed) -> AND
(Completed) -> SHL
(Completed) -> SHR

OpenArray [^]
Backlinks: ParameterType

(Completed) -> ARRAY OF QualifiedIdent
(Completed) -> ARRAY OF STRING
(Completed) -> ARRAY OF FILE
(Completed) -> ARRAY OF CONST

Package [^]
Backlinks: Goal

(Completed) -> PACKAGE QualifiedIdent ';'
(Completed) [RequiresClause]
(Completed) [UsesClause]
(Completed) (AssemblyAttribute)*
(Completed) END '.'

PackedType [^]
Backlinks: Type

(Completed) -> PACKED Type

Parameter [^]
Backlinks: MethodHeading, ProcedureType, Property

(Completed) -> [VAR | CONST | OUT]
(Completed) IdentList
(Completed) [':' ParameterType]
(Completed) ['=' Expression]

ParameterExpression [^]
Backlinks: Atom

(Completed) -> Expression [':' Expression [':' Expression]]

ParameterType [^]
Backlinks: Parameter

(Completed) -> QualifiedIdent
(Completed) -> STRING
(Completed) -> FILE
(Completed) -> OpenArray

ParenthesizedExpression [^]
Backlinks: Particle

(Completed) -> '(' Expression ')'

Particle [^]
Backlinks: Atom

(Completed) -> <number>
(Completed) -> <stringliteral>
(Completed) -> Ident
(Completed) -> NIL
(Completed) -> ParenthesizedExpression

(Completed) -> SetLiteral
(Completed) -> STRING
(Completed) -> FILE

PointerType [^]

Backlinks: Type

(Completed) -> '**^**' Type

PortabilityDirective [^]

Backlinks: ConstantDecl, Directive, FieldDecl, TypeDecl, Unit, VarDecl

(Completed) -> platform
(Completed) -> deprecated
(Completed) -> library
(Completed) -> experimental

ProcedureType [^]

Backlinks: Type

(Completed) -> (PROCEDURE | FUNCTION)
(Completed) ['**(**' (Parameter ['**;**'])* '**)**']
(Completed) ['**:**' MethodReturnType]
(Completed) (Directive)*
(Completed) [OF OBJECT]
(Completed) (Directive)*

Program [^]

Backlinks: Goal

(Completed) -> (PROGRAM | LIBRARY) Ident ['**(**' IdentList '**)**'] '**;**'
(Completed) -> [UsesClause]
(Completed) (ImplementationDecl)*
(Completed) InitSection '**.**'

Property [^]

Backlinks: MethodOrProperty

(Completed) -> [CLASS]
(Completed) PROPERTY Ident
(Completed) ['**[**' (Parameter ['**;**'])+ '**]**']
(Completed) ['**:**' MethodReturnType]
(Completed) (PropertyDirective)*
(Completed) '**;**'

PropertyDirective [^]

Backlinks: Property

(Completed) -> '**;**' DEFAULT
(Completed) -> DEFAULT Expression
(Completed) -> DISPID Expression
(Completed) -> IMPLEMENTS (QualifiedIdent ['**,**'])+
(Completed) -> INDEX Expression
(Completed) -> NODEFAULT
(Completed) -> READ Expression
(Completed) -> READONLY
(Completed) -> STORED Expression
(Completed) -> WRITE Expression
(Completed) -> WRITEONLY

QualifiedIdent [^]

Backlinks: ClassHelperType, ClassOfType, ClassType, ExceptionItem, FileType, InterfaceType, MethodHeading, MethodReturnType, OpenArray, Package, ParameterType, PropertyDirective, RecordHelperType, RequiresClause, TypedConstant, VariantSection

(Completed) -> Ident ('**.**' ExtendedIdent)*

RaiseStatement [^]
Backlinks: SimpleStatement

(Completed) -> RAISE [Expression [AT Expression]]

RecordHelperType [^]
Backlinks: Type

(Completed) -> RECORD HELPER FOR QualifiedIdent
(Completed) (VisibilitySection)*
(Completed) END

RecordType [^]
Backlinks: Type

(Completed) -> RECORD
(Completed) (VisibilitySection)*
(Completed) [VariantSection]
(Completed) END

RelOp [^]
Backlinks: Expression

(Completed) -> '='
(Completed) -> '>'
(Completed) -> '<'
(Completed) -> '<='
(Completed) -> '>='
(Completed) -> '<>'
(Completed) -> IN
(Completed) -> IS
(Completed) -> AS

RepeatStatement [^]
Backlinks: SimpleStatement

(Completed) -> REPEAT [StatementList] UNTIL Expression

RequiresClause [^]
Backlinks: Package

(Completed) -> REQUIRES (QualifiedIdent [','])+ ';'

SetLiteral [^]
Backlinks: Particle

(Completed) -> '[' [ExpressionOrRangeList] ']'

SetType [^]
Backlinks: Type

(Completed) -> SET OF Type

SimpleExpression [^]
Backlinks: Expression, ExpressionOrRange

(Completed) -> Term (AddOp Term)*

SimpleStatement [^]
Backlinks: Statement

(Completed) -> BareInherited
(Completed) -> ExpressionOrAssignment
(Completed) -> GotoStatement
(Completed) -> Block
(Completed) -> IfStatement
(Completed) -> CaseStatement
(Completed) -> RepeatStatement

(Completed) -> WhileStatement
(Completed) -> ForStatement
(Completed) -> WithStatement
(Completed) -> TryStatement
(Completed) -> RaiseStatement

Statement [^]

Backlinks: CaseSelector, ExceptionItem, ForStatement, IfStatement, StatementList, WhileStatement, WithStatement

(Completed) -> LabelId ':' [SimpleStatement]
(Completed) -> SimpleStatement

StatementList [^]

Backlinks: Block, CaseStatement, InitSection, RepeatStatement, TryStatement

(Completed) -> ([Statement] [';'])+

StringType [^]

Backlinks: Type

(Completed) -> STRING
(Completed) -> STRING '[' Expression ']'

Term [^]

Backlinks: SimpleExpression

(Completed) -> Factor (MulOp Factor)*

TryStatement [^]

Backlinks: SimpleStatement

(Completed) -> TRY
(Completed) [StatementList]
(Completed) (FINALLY [StatementList]
(Completed) | EXCEPT (
(Completed) [StatementList] |
(Completed) (ExceptionItem)* [ELSE [StatementList]]
(Completed))
(Completed) END

Type [^]

Backlinks: ArrayType, ConstantDecl, FieldDecl, PackedType, PointerType, SetType, TypeDecl, VarDecl

Note: Delphi assumes that a Type starting with '(' is an enum, not an expression.

(Completed) -> EnumeratedType
(Completed) -> ExpressionOrRange
(Completed) -> ArrayType
(Completed) -> SetType
(Completed) -> FileType
(Completed) -> RecordHelperType
(Completed) -> RecordType
(Completed) -> PointerType
(Completed) -> StringType
(Completed) -> ProcedureType
(Completed) -> ClassHelperType
(Completed) -> ClassOfType
(Completed) -> ClassType
(Completed) -> InterfaceType
(Completed) -> PackedType

TypedConstant [^]

Backlinks: ConstantDecl, TypedConstant, VarDecl

(Completed) -> Expression
(Completed) -> '(' (QualifiedIdent ':' TypedConstant [';'])+ ')'
(Completed) -> '(' (TypedConstant [','])+ ')'

(Completed) -> '**(**' ')'

TypeDecl [^]
Backlinks: TypeSection

(Completed) -> Ident '**=**' [TYPE] Type (PortabilityDirective)* '**;**'
(Completed) -> Ident '**=**' CLASS '**;**'
(Completed) -> Ident '**=**' DISPINTERFACE '**;**'
(Completed) -> Ident '**=**' INTERFACE '**;**'

TypeSection [^]
Backlinks: ImplementationDecl, InterfaceDecl, VisibilitySectionContent

(Completed) -> TYPE (TypeDecl)+

UnaryOperator [^]
Backlinks: Factor

(Completed) -> NOT
(Completed) -> '**+**'
(Completed) -> '**-**'
(Completed) -> '@'
(Completed) -> INHERITED

Unit [^]
Backlinks: Goal

(Completed) -> UNIT Ident (PortabilityDirective)* '**;**'
(Completed) InterfaceSection
(Completed) ImplementationSection
(Completed) InitSection '**.**'

UsedUnit [^]
Backlinks: UsesClause

(Completed) -> Ident
(Completed) -> Ident IN <stringliteral>

UsesClause [^]
Backlinks: ImplementationSection, InterfaceSection, Package, Program

(Completed) -> (USES | CONTAINS)
(Completed) (UsedUnit ['**,**'])+ '**;**'

VarDecl [^]
Backlinks: VarSection

(Completed) -> IdentList '**:**' Type
(Completed) (PortabilityDirective)*
(Completed) [ABSOLUTE Expression | '**=**' TypedConstant]
(Completed) (PortabilityDirective)*
(Completed) '**;**'

VariantGroup [^]
Backlinks: VariantSection

(Completed) -> ExpressionList '**:**'
(Completed) '**(**'
(Completed) -> (FieldDecl)*
(Completed) [VariantSection]
(Completed) '**)**' ['**;**']

VariantSection [^]
Backlinks: RecordType, VariantGroup

(Completed) -> CASE [Ident '**:**'] QualifiedIdent OF
(Completed) (VariantGroup)+

VarSection [^]
Backlinks: ImplementationDecl, InterfaceDecl

(Completed) -> (VAR | THREADVAR) (VarDecl)+

Visibility [^]
Backlinks: VisibilitySection

(Completed) -> STRICT PRIVATE
(Completed) -> STRICT PROTECTED
(Completed) -> PRIVATE
(Completed) -> PROTECTED
(Completed) -> PUBLIC
(Completed) -> PUBLISHED

VisibilitySection [^]
Backlinks: ClassHelperType, ClassType, RecordHelperType, RecordType

(Completed) -> [Visibility]
(Completed) (VisibilitySectionContent)*

VisibilitySectionContent [^]
Backlinks: VisibilitySection

(Completed) -> FieldSection
(Completed) -> MethodOrProperty
(Completed) -> ConstSection
(Completed) -> TypeSection

WhileStatement [^]
Backlinks: SimpleStatement

(Completed) -> WHILE Expression DO [Statement]

WithStatement [^]
Backlinks: SimpleStatement

(Completed) -> WITH ExpressionList DO [Statement]